USING GENETIC ALGORITHMS TO SOLVE LAYOUT OPTIMISATION PROBLEMS IN RESIDENTIAL BUILDING CONSTRUCTION

Dr Andy Connor
Software Engineering Research Lab, AUT
Auckland, New Zealand
andrew.connor@aut.ac.nz

Mr Wilson Siringoringo
Software Engineering Research Lab, AUT
Auckland, New Zealand
wilson.siringoringo@aut.ac.nz

Abstract

This paper outlines an approach for the automatic design of material layouts for the residential building construction industry. The goal is to cover a flat surface using the minimum number of rectangular stock panels by nesting the off cut shapes in an efficient manner. This problem has been classified as the Minimum Cost Polygon Overlay problem. Results are presented for a typical problem and two algorithms are compared.

1 INTRODUCTION

This paper describes the application of Genetic Algorithms to a class of layout optimisation problems found in the construction industry, with particular relevance to the construction of residential buildings. This class of problem has been defined by previous work as the Minimum Cost Polygon Overlay (MCPO) problem [1].

The aim of the work is automate the sheet layout process for flat sections of a building. The goal of such an automated process is to construct a solution that allows the sections to be completely covered with the optimum layout. This can be defined as the smallest possible amount of stock material, which is cut with minimum amount of effort. It is also important for these optimum solutions to be found in a reasonable amount of time.

2 2D LAYOUT OPTIMISATION

Optimum two-dimensional layout is a class of problems encountered in many industries. The problems are characterized with the need to pack non-overlapping shapes in an enclosed plane with the aim of minimizing the area outside the boundaries of the shapes, therefore maximizing the utilization of the material in the base sheet.

The actual optimum two-dimensional layout problem exists in several variants. Dyckhoff [2] makes an attempt to provide a systematic classification of such optimization problems. He uses the term cutting and packing (C&P) as a generic name for the problem and all its variants. Amongst these variants are the *sheet layout* problem, *bin packing* and *strip packing* problems, *optimum floor plan* problem, and *cutting stock* problem.

Strip packing (SP) and bin packing (BP) are specific subclasses of the generic sheet layout problems, with the objective limited to placing rectangular items within fixed width container. Furthermore, rotation is allowed only at 90º increments whereas mirroring is irrelevant because of the rectangle's symmetry. The subject of SP and BP covers problems of various dimensions. However, two-dimensional BP (2BP) and SP (2SP) problems can be considered a subset of sheet layout problem class [3].

A rather unique variant of the optimum two-dimensional layout problem is found in the construction industry, namely the MCPO problem. A polygon shaped area such as wall or ceiling is to be tiled with covering sheet material such as cardboard or plywood. With such tiling, it is essential that the entire surface is covered with no gaps or overlaps. The panels are obtained from the supplier in fixed size rectangles. Typically the individual panel is much smaller than the area to be covered. It is also anticipated that the enclosing area may have an irregular outline.

The problem is demonstrated in Figure 1. To keep the construction expenses under control, the builder must arrange the panels in a way that keeps the cost variables low. Such parameters include the number of panels allocated, the amount of discarded off cuts, and the amount of effort required for cutting the panels.

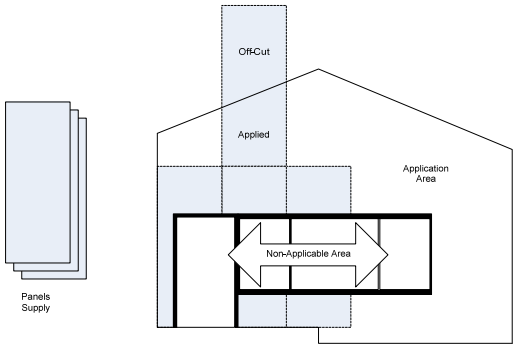

Figure 1: Wall Overlay with Fixed Size Panels

A similar problem has been encountered in the shipbuilding industry, particularly in cutting steel sheets to cover various parts of the ship [4].

When the panel is homogenous, such as with sheet metal, it is desirable to reuse the off cuts to cover irregular regions at other places, as this has the potential to reduce the total number of sheets required. A particular example

was made by Sibley-Punnett and Bossomaier [5] regarding the reuse of off cuts from corrugated iron roofs. The justification for such effort is provided by the high cost of delivering the roofing material.

The diversity of materials used for constructing a building provides no guarantee that such homogeneity exists for materials used for a particular area. The implication is that the constraints for a particular section of the building cannot be predetermined. In response, a computer program used to resolve such problem must be capable of finding the solution under a varying set of constraints to allow it to be used for any specific instance of the general problem.

Closer examination reveals that the MCPO problem is composed of two sub-problems which must be resolved sequentially, although each sub-problem still belongs to a class of two-dimensional layout optimization problem. For a given enclosed area and given dimensions of rectangular panels, the requirement is twofold:

(i) Find the optimum arrangement of whole panels in which the covered area within the enclosure is maximized. The by-product of this process is a set of irregular shapes which represent the remaining exposed areas.

(ii) Resolve how such irregular shapes can be nested within the minimum number of panels. Shapes that are bigger than the panel itself are cut at angles parallel with the rectangle's axes to allow such nesting.

This decomposition into two sub-problems can potentially mask the complexity of the task of finding the optimum solution. It is important to recognize that in the construction industry, the actual size of the panels is in itself a design parameter. In some applications, the panel size will remain fixed for the two sub-problems whilst for other applications the panel size could potentially be varied. With this in mind, it becomes apparent that the problem is complex with potentially many locally optimum solutions.

3 GENETIC ALGORITHMS

Genetic Algorithms are one of many heuristic approaches that have been developed for solving complex optimisation problems and dealing with the existence of multiple optima in a problem solution space. These algorithms use the concept of a population of individuals which are subject to a series of probabilistic operators such as *mutation*, *selection* and *recombination*. Each individual represents a potential solution to a given optimization problem. During the computation process, the population will undergo a draconian process in which stronger individuals thrive whilst the weaker ones perish.

Goldberg [6] asserts that GAs are more robust than many other optimization techniques, particularly when the search space contains many local optima. He further attributes the robustness of GAs to four special characteristics of the algorithm:

(i) Instead of working directly with the optimization parameters, GA works with a coded set of the parameters.

(ii) The optimization result is obtained from a population of points instead of a single point.

(iii) GAs directly use the objective function to calculate the payoff information instead of derivatives or other auxiliary information.

(iv) Probabilistic transition rules are used in GAs instead of deterministic rules.

GAs have been applied to a wide range of problems that have been considered intractable to other approaches and as a result have been selected as a candidate solution for solving layout optimisation problems in the construction industry.

In many GA implementations in the literature the chromosome is commonly implemented as a finite-length binary vector. A binary vector provides the maximum flexibility for parameter coding and interpretation in much the same way as basic data types such as numerical or symbolic values are internally represented in the computer memory. Non-binary strings are also used however, in specific cases such as when representing nodes in Traveling Salesman Problem (TSP), where a binary equivalent is impractical or inefficient [7].

Because of its very flexibility, coding the optimization parameters into a gene string can be a daunting task. For any given optimization problem, there are typically a number of possible ways to code the parameters into the gene string, some are better than others. There is surprisingly little available literature providing a general guideline for coding GA parameters. Coding guidelines for specific domains do exist however, such as those proposed by Nagao for optimization of numerical parameters [8].

The three basic operators in evolutionary computing, *mutation*, *selection* and *recombination*, are used in the implementation of the genetic algorithm. Specifically in the context of GAs, the operators are referred to respectively as *mutation*, *reproduction*, and *crossover* [7].

3.1 GA Parameter Coding

Parameter coding for GAs has a major contribution towards the effectiveness of the optimization engine. A set of chromosomes containing wrong sets of parameters

or poorly mapped parameter values will ruin an otherwise good GA implementation. Similarly a good representation of the parameters will make it possible for the GA implementation to realize its full potential.

Parameter coding is especially problematic in the MCPO problem under consideration. This is especially true for the second-stage of the optimization, because interdependencies exist among the parameters. The second stage optimization appears to be best modelled on the 2BP problem. Although the use of a GA in solving 2BP can be found in a number of publications [9][10][11][12], none provides the technical description about the actual parameter coding. Perhaps the most technical detail can be found in the work of Shian-Miin, Cheng-Yan, & Jorng-Tzong [13] where complex tree structures are used to represent the nested objects.

In the absence of an exact description regarding the parameter coding of 2BP optimization, a novel solution for parameter coding has been devised. Substantial effort has been expended in designing the chromosome. Not only because there are multiple parameters involved in layout optimization problems, but some of the parameters are also inter-dependent. To construct a suitable model, it is quite worthwhile to examine the parameters that define a second-stage solution in MCPO. Such parameters are:

(i) The total number of stock panels required

(ii) The list of pieces that are nested within each stock panel

(iii) The placement coordinates of each piece within a stock panel

(iv) The rotation and flipping applied to that particular piece

Evidently the first parameter is dependent on the second parameter. Similarly the second parameter is largely dependent on the third and fourth parameters. In the face of this, the only information available to determine the value of those parameters is the list of irregular panels represented by their vertices. This all leads to a situation radically different from standard sheet layout problems found in the literature.

To reiterate, in standard sheet layout problems commonly found in the literature, only a *single* container is provided. The solution designer is therefore allowed to use the chromosome to directly represent the container and map the genes within the chromosome to the nested pieces. Static blocks of bits can be used to represent the placement coordinates of each piece, its rotation, and so on.

This static mapping cannot be easily applied to MCPO, since the number of containers itself is a variable to begin with. The only possible way to accommodate all the parameters within a single chromosome using a static mapping is by allocating a large block of bits for each stock panel to make it able to contain *all* the pieces, and ensure that enough stock panel blocks are provided within that single chromosome to anticipate the possibility of having only one piece per panel. Unsurprisingly, the resulting bit string is very large and prohibitively inefficient to be implemented.

A much more feasible solution is to deliberately use only a few parameters in the main model, and to relegate the task of populating the rest of the parameters somewhere else. Since the first two parameters identified above are the most crucial, they are selected to be represented in the chromosome.

Resolving the third and fourth parameters is important to determine whether the solution for first and second parameters is legal. It is most appropriate to make finding their correct values an integral part of the fitness evaluation function for the original chromosome.

This is done by utilizing the same sequential placement routines as used in the greedy algorithm which has been which has been adopted as it is fast and deterministic in nature. It is important to not that GAs are typically used to implement a *simultaneous placement* nesting strategy. The fact that all nesting optimization algorithms implemented in this project eventually use a *sequential placement* strategy rules out the possibility of comparing the performances of the two.

After all the relevant decisions been made as discussed above, the problem is now sufficiently reduced to enable the actual modeling of the chromosome. There are only two parameters remaining to be coded in the chromosome:

(i) The total number of stock panels required

(ii) The list of pieces that are nested within each stock panel

Direct coding to the genes in the chromosome is still not possible because the second parameter is of a variable length. To solve this problem, indirect coding employing the concept of *clusters* is used.

In this technique, static blocks in the chromosome are mapped to the pieces to be nested. This represents the worst case solution, where each piece requires an individual stock panel to be used. From the first step of the solution, it is known that all pieces to be nested are smaller than the stock panels therefore this provides an upper threshold for the maximum number of panels required. Each panel is associated with a fixed-width block of bits in the chromosome. This block contains only a single variable of integer type, namely the cluster ID.

Figure 2 shows the association between the panels and the blocks in the chromosome.

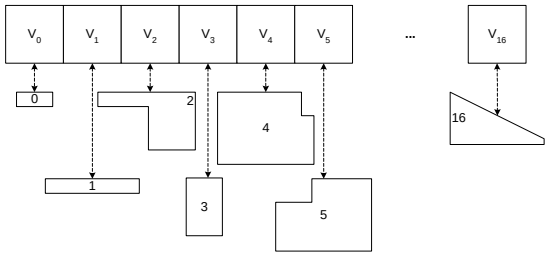

Figure 2: Gene to Panel Mapping

The value of each variable points to an imaginary cluster to which the panel belongs. Figure 3 shows an example of a populated chromosome with the imaginary clusters that result. Because only 17 panels exist, the binary string can use five bits to hold the cluster ID.

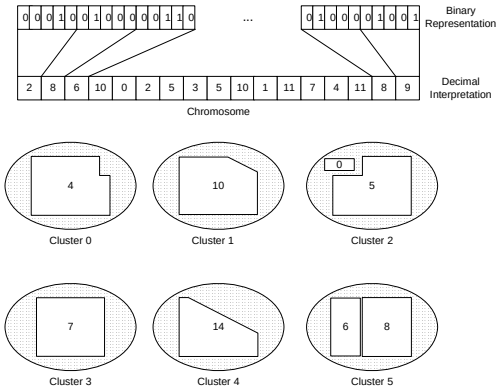

Figure 3: Interpreting a Candidate Chromosome

Using Figure 2 as reference, it is easy to decode the chromosome to find that the Panel 0 is a member of Cluster 2, whereas Panel 1 is a member of Cluster 8, and so on. Similarly, Cluster 0 appears to have only a single member, i.e. Panel 4, whereas Cluster 2 has two members: Panel 0 and Panel 5.

A cluster is regarded as legal if all its members can be nested in a single stock panel. As previously discussed, part of the fitness function's task is to discover whether such nesting is possible. In the case of an invalid cluster being encountered, there are a number of possible ways to respond.

The use of clusters effectively addresses the variable length problem of the nesting list. Because the list only exists implicitly in the chromosome, no assumption about the number of clusters needs to be made beforehand. Furthermore by allowing the pieces to map themselves to the clusters, it is guaranteed that the number of clusters will always be less than or equal to the number of pieces.

Typically, the number of bits allocated for each panel is a good deal more than required to express all the possible Cluster IDs for a given optimization problem. Consequently, assigning the pieces with a random Cluster ID number will often result in single-member clusters with widely scattered IDs. While this phenomenon does not affect the validity of the result, it does potentially bias the optimization engine into giving an inefficient result. This problem is easy to solve however, by using a *modulo* operator to convert all IDs to the acceptable range.

3.2 Valid/Invalid Chromosomes

A chromosome in the context of layout optimization is accepted as valid only when all pieces can be successfully nested in their associated stock panel. Its opposite is the invalid chromosome, which contains one or more clusters whose members cannot be nested in a stock panel. Because the search is set-oriented, there is no guarantee that all the clusters extracted from a chromosome are valid. Invalid clusters are found very frequently in the actual tests because many of the individual pieces are quite large compared to the size of stock panels, invariably claiming most of the available area after only one or two nested pieces.

Mindlessly discarding invalid chromosome is not a desirable option using a GA. Because the direction of the search is dictated by the collective patterns in its population of chromosomes, great care must be taken to ensure that the population can survive and retain good quality patterns at each turn of generation. A dilemma inevitably arises: should an invalid chromosome be retained in spite of its lack of value as a solution; or should it be discarded and risk the population dwindling and becoming stagnant after just a few generations? The sensible answer must lie somewhere between those two extremes.

Whilst a number of strategies have been considered [1], at this stage an approach labelled *Redistribute from Beginning* (RFB) has been adopted. An attempt is made at nesting the rejected piece in an already created panel before creating a new cluster at the end of the list if required. Theoretically this approach will result in a more even distribution of the pieces, and ultimately a better overall fitness value.

3.3 Cluster Placement Strategy

An important aspect of generating clusters that are both valid and good is the positioning of the pieces in an available container. The second problem to be solved about a particular piece is about where it should be placed within the container. It is evident that when the container is considered continuous, the candidate panel may be placed inside in an infinite number of ways.

Reducing the container to a discrete set of possible placement choices is vital to make search possible. Given the exponential nature of the size of the overall optimization problem as a whole, limiting the number of possible ways of placing candidate panels in the container from that discrete set is also necessary. This particular implementation uses *incident vertex* placement, which is an approach similar to linear programming. If the area of the container is considered as the feasible area, then the potential optimum solutions are associated to its vertices. Only those vertices will be evaluated as incident vertex candidates for the panel at hand. The panel is then shifted to various places to make its vertices overlap with those of the container.

Figure 4 shows the evaluation of how a small triangular piece can be placed inside a rectangular container using such a method. It appears that twelve possible solutions exist, of which three are valid as a nesting solution.

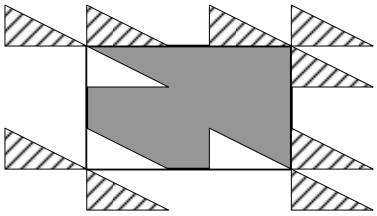

Figure 4: Layout Solution by Vertex Incidence

Because the solution is not singular, a further decision must be made to select the "best" from these equally valid options. There are two options available in response: those based on the *first fit* and the *best fit* strategies. The results presented in this paper are based on the use of the best fit strategy. Figure 5 shows three legal ways a triangle *abc* can be placed inside a rectangular container. These three candidates will be evaluated to determine which one is the "best". The notion of best solution is elusive and problem-specific however, requiring analysis about what goal the algorithm is set to achieve and what means are available to achieve it.

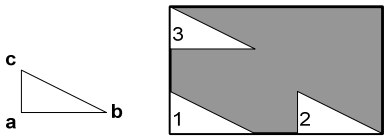

Figure 5: Candidate Solutions for Best Fit Placement

Because the objective of layout optimization is to put the pieces so as to occupy as much container space as possible without overlapping, the logical posture of the best fit strategy is to maintain a continuous and convex free space after each piece is placed. Hence, the best solution for a given iteration is the one that provides the least possible obstructions in the remaining unoccupied space.

With the criterion of the best solution established, the next task is to develop an effective and inexpensive way to make the necessary evaluation. Unfortunately there is no straightforward way the amount of obstruction within the vacant space can be measured. A less direct calculation based on *overlapping edges* is used instead. For a given candidate solution, the length of the edges of the piece that overlap with the outline of the container is calculated. If previously placed pieces exist, the length of overlapping edges with those pieces is also added. The best solution is defined as the one with highest total length of the overlapping edges.

4 RESULTS

A number of experiments have been conducted that compare the performance of the GA against a simple implementation of Greedy algorithm that utilises the same best fit strategy. The Greedy approach cycles through the available pieces and attempts to fit the largest piece in the available space of an existing panel. If the piece does not fit in any of the existing panels, a new panel is used to fit the piece. One such experiment was conducted on the complex roof problem. The roof is shown in Figure 6.

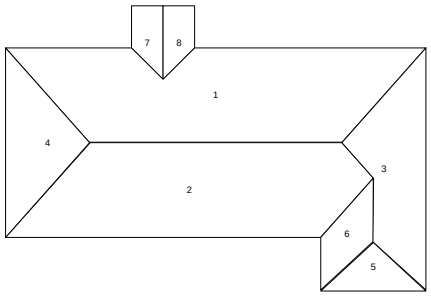

Figure 6: Complex Roof Layout

Table 1 compares the performance of the GA against the Greedy search method. In this experiment, the same best fit strategy was used and the material was allowed to rotate 180° only.

Criteria	Greedy	GA
Full Panels Used	25	25
No. Offcuts to Nest	129	129
Total No. Pieces	154	154
Stock Panels Used	64	85
Shared Edge Length	9930	9814
Area to be Covered	124153	124153
Area of Stock Panels	128000	170000
Wasted Material	3847	45847
Solution Efficiency	97%	73%
Search Duration	0:00:03	0:45:35

Table 1: Solution Quality Criteria

Whilst a number of different experiments have been undertaken, these results provide some interesting insight into the approach taken. Despite the Genetic Algorithm being a more robust algorithm, it has consistently found less attractive solutions than the simple Greedy algorithm.

Retrospective analysis of the GA implementation has identified the potential cause of this unexpected poor performance. Whilst the mapping of pieces to clusters is a powerful approach for dealing with the fact that the number of pieces is in fact an optimisation parameter, the method of dealing with invalid clusters has a certain weakness.

The redistribution of pieces in an invalid cluster is also powerful, however the weakness is the new cluster data is not reintroduced into the GA population. The GA is therefore operating on a large population of invalid clusters, which could be limiting the performance of the method.

5 CONCLUSIONS

This paper has described an approach for automating a class of 2D material layout optimisation defined as the Minimum Cost Polygon Overlay problem. This class of problem is common in the construction industry where large areas are required to be fully covered using the minimum number of rectangular stock panels.

The key element of this problem is the allocation of irregular shapes to multiple stock panels and the reuse of off cut sections to minimise wastage. Two algorithms have been applied to the solution of this stage, namely a deterministic algorithm based on the allocation of the next available piece, and a heuristic algorithm for a pseudo-simultaneous approach.

The deterministic, or Greedy, algorithm clearly outperforms the heuristic approach which utilises a Genetic Algorithm. Analysis of the implementation has show that the approach for dealing with invalid clusters in the Genetic Algorithm is the most likely cause for such poor performance.

Whilst the results of the Greed algorithm are suitable for use by architects and builders working in this area, future work will investigate improved approaches for dealing with invalid clusters and also investigate the suitability of alternative heuristic algorithms.

6 ACKNOWLEDGEMENTS

This research has been supported by Technology New Zealand through the Technology for Industry Fellowships scheme (grant number BISC0502) and this assistance is gratefully acknowledged.

7 REFERENCES

[1] Author 2. (2007). Minimum Cost Polygon Overlay with Rectangular Shape Stock Panels. Masters Thesis, Awarding University.

[2] Dyckhoff, H. (1990). Typology of cutting and packing problems. European Journal of Operational Research, 44(2), 145-159.

[3] Lodi, A., Martello, S., & Monaci, M. (2002). Two-dimensional packing problems: A survey. European Journal of Operational Research, 141(2), 241-252.

[4] Adamowicz, M., & Albano, A. (1976). Nesting Two-Dimensional Shapes In Rectangular Modules. IEEE Transactions on Systems, Man and Cybernetics, 8(1), 27-33

[5] Sibley-Punnett, L., & Bossomaier, T. (2001). Optimisation techniques for roof layout. Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology ('TENCOM')

[6] Goldberg, D. E. (1989). Genetic Algorithm in Search, Optimization, and Machine Learning. Reading, Massachusetts: Addison-Wesley Publishing

[7] Ansari, N., & Hou, E. (1997). Computational Intelligence for Optimization. Newark, New Jersey: Kluwer Academic Publishers.

[8] Nagao, T. (1996, 20-22 May 1996). Homogeneous Coding for Genetic Algorithm Based Parameter Optimization. Paper presented at the Proceedings of the IEEE Conference on Evolutionary Computation, Nagoya, Japan

[9] Chan, F. T. S., Au, K. C., & Chan, P. L. Y. (2005). A genetic algorithm approach to bin packing in an ion plating cell. Proceedings of the Institution of Mechanical Engineers, Part B (Journal of Engineering Manufacture), 219(B1), 1-13

[10] Falkenauer, E., & Delchambre, A. (1992). A genetic algorithm for bin packing and line balancing, Paper presented at the 1992 International Conference on Robotics and Automation, Nice, France

[11] Lewis, J. E., Ragade, R. K., Kumar, A., & Biles, W. E. (2005). A distributed chromosome genetic algorithm for bin-packing. Robotics and Computer-Integrated Manufacturing, 21(4-5), 486-495

[12] Liu, D., & Teng, H. (1999). Improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research, 112(2), 413-420

[13] Shian-Miin, H., Cheng-Yan, K., & Jorng-Tzong, H. (1994). On solving rectangle bin packing problems using genetic algorithms, Paper presented at the 1994 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, USA